
# Survey of Congestion Control Techniques For an ATM Network

*Duke Hong, Tatsuya Suda and Jaime Jungok Bae*

Technical Report No. 91-48

Department of Information and Computer Science
University of California, Irvine
Irvine, CA 92717

# SURVEY OF CONGESTION CONTROL TECHNIQUES FOR AN ATM NETWORK [1]

DUKE HONG, TATSUYA SUDA, JAIME JUNGOK BAE

Department of Information and Computer Science, University of California, Irvine, CA 92717

**Abstract**

The emerging broadband integrated services digital network is expected to adopt ATM (Asynchronous Transfer Mode) as the transport network. This new network must support several classes of service with varying delay and loss requirements. It must also operate with link speeds in the hundreds of megabits per second and be scalable up to potential link speeds on the order of gigabits per second. The requirements to support multiple services and high speed make the congestion control in an ATM network difficult. This paper reviews some of the techniques for prevention and control of congestion in an ATM network.

## 1 Introduction

Broadband integrated services digital network (B-ISDN) is expected to be introduced into commercial service by the mid-1990's. It should provide support for transmission of both asynchronous data and synchronous real-time traffic on a single transmission network. It should also provide such diverse services as interactive and distributive services, broadband and narrowband rates, support for bursty and continuous traffic, connection oriented and connectionless calls, and point-to-point and complex communications [1]. To support these services, broadband ISDN requires large amounts of bandwidth available only through fiber optics. With the advent of fiber optics, transmission speeds have already reached into the hundreds of megabits per second and are expected to go higher.

Currently, ATM is seen as the target transport method for B-ISDN because of its ability to handle future services. ATM is a packet[2] (or cell) switched network. Since bandwidth is allocated on demand, new services can be readily adapted. Another factor which favors ATM is the bursty nature of calls expected. Bursty calls are calls which generate a large amount of traffic at some high peak rate for a short period of "active" time and generate little or no traffic for some "idle" time. If bandwidth were allocated based on the peak rate (*deterministical multiplexing*), the capacity of the network would be wasted when the call is idle. In ATM, bursty calls are *statistically multiplexed*; each call is assigned some bandwidth that is lower than its peak bit rate. Statistical multiplexing is more bandwidth efficient and allows more calls to enter the network, especially when a large portion of the traffic is bursty. Because of these advantages, ATM is expected to be adopted by CCITT as the transport method for B-ISDN. For more on ATM, see [1-6].

[1]This material is based upon work supported by the National Science Foundation under Grant No. NCR-8907909. This research is also in part supported by University of California MICRO program.

[2]Packets on an ATM level are called "cells".

## 2 The congestion control problem

Congestion control in an ATM based B-ISDN is difficult because of the high link speed, diverse service requirements, and diverse characteristics of the traffic ATM is expected to support. One effect of a high speed channel is that cell processing time is increased relative to the cell transmission time. For instance, at 150 Mb/s link speed and with an ATM cell size of 53 bytes, cells must be switched at a rate greater than (53 bytes per cell)(8 bits per byte)/(150 Mb/s) = one cell per 3 $\mu$s. It is apparent that schemes used to control cells in ATM must be performed in hardware to avoid the excessive processing time of software. Therefore it is extremely important that protocols not only be simple enough for implementation in hardware, but they must also be simple enough to support these sub-microsecond switching speeds.

Another problem caused by the high speed is the increased propagation-bandwidth product. This is the amount of traffic that can be in transit during a propagation delay time. Consider a 1Gb/s line with a propagation delay of 20 msec for a cross-continental distance of 6000 km. In existing networks, when a node becomes congested, it can send a choke packet to the senders to stop or slow transmission. By the time that the receiver sends a control message to throttle the transmitter, 20 megabits are already in transit. By the time that the control message reaches the transmitter, 40 megabits will have to be retransmitted causing a significant reduction of throughput. This can make many of the current congestion control schemes ineffective. As shown above, such reactive form of control is generally accepted as inappropriate in the ATM environment.

The second issue is handling multiple service requirements. Some services such as voice, real-time video, and data for real-time control have strict delay requirements, whereas some services such as file transfers have strict loss requirements. In ATM, even if a call is admitted to the network, the network delay and cell loss may not be guaranteed. To satisfy these diverse service requirements, "favorable" treatment of some service classes may be necessary. The common ways of dealing with service classes are with priority treatment of cells. Because of the page limit, priority schemes are not included in this paper. For further information on priority schemes, refer to [2,5].

The third issue that affects congestion control in ATM is the diverse traffic characteristics with different degrees of burstiness. As discussed earlier, bursty traffic generates traffic at a large peak rate for some short period of time and generates little or no traffic for some longer time. Bursts can accumulate in buffers and cause periodic buffer overflow which in turn results in excessive delays and cell loss. In ATM, to allow statistical multiplexing, bursty calls should only be allocated some bandwidth less than the peak rate so as to take advantage of their burstiness. Determining how much bandwidth to allocate to a bursty call must be resolved. In addition, a shaping function, such as a leaky bucket, can be utilized to smooth the input traffic by altering some traffic characteristics like the minimum cell interarrival time and burst peak rate to reduce some of the effects of burstiness.

In summary, the following important issues in ATM congestion control remain unresolved. What effect does the increased propagation-bandwidth product and fast cell processing time have

on the choices made on congestion and flow control? How do we determine how much bandwidth to allocate to a bursty call? What is the role of burstiness in deciding a congestion control scheme? If bursty traffic is detrimental to network performance, what steps can we take to shape the input traffic so that it does not adversely affect the network? In the following sections, some of the recent papers that have directed these issues are reviewed.

# 3 Admission control

Reactive control of congestion is unsuitable for ATM because of the long propagation delay. Prevention of congestion can avoid situations where the network must react to congestion. Admission control is an effective form of preventive control. Admission control can prevent congestion by limiting the number of calls in the network to a level where the network's resources are adequate to maintain throughput guarantees.

## 3.1 Equivalent bandwidth

Admission control is based on allocation of resources. When a new call arrives at an ATM network, the call is admitted as long the network can support the *expected* traffic. Since ATM is packet-switched, the network does not explicitly reserve any bandwidth for calls. Instead, expected bandwidth use per call is used as indicators of the expected volume of traffic to determine if the network can adequately support more calls. The difficulty is in determining the amount of bandwidth that calls are expected to require.

If the network only supports CBO traffic, then determining the expected traffic is an easy process. The expected traffic is equivalent to the peak bit rate of the call since any fluctuations from the peak will be minor. For bursty calls, the network can better utilize the available bandwidth by statistically multiplexing calls. To statistically multiplex, the network must determine a bandwidth requirement for a call that is less than the peak bit rate. Determining bandwidth requirement should take into account factors such as a call's burstiness ratio $Peak\_bit\_rate/Average\_bit\_rate$, burst length, service requirement, burst interarrival time, etc. The expected amount of bandwidth that a call uses is called the *equivalent bandwidth* of a call.

The measure of equivalent bandwidth is dependent on several factors. The most important factor is the ratio of the peak bit rate of the call and the link rate [7]. It was shown in [8] that (for homogeneous sources) for increasing values of available link rate, the multiplexing gain increases and the equivalent bandwidth to allocate decreases. Calls with high values of peak to link ratios, when the two rates differ by about one order of magnitude (0.1) or less, required peak bandwidth allocation [7-9]. It was shown in [10] that the network could accept 50% more bursty traffic when the peak to link ratio is 1/1000 than when the ratio is 1/10. If statistical multiplexing is possible (low peak to link ratio), then the burstiness was shown in [7] to be the most important factor. Sensitivity to burst length is also especially important when the burst length is near the buffer

size. This is because one burst may overflow the buffer causing cell loss and delays. Other factors in determining equivalent bandwidth include the mean bit rate, cell loss and delay requirements of calls, and the mix of services used in the network [8].

During call admission, the equivalent bandwidth is used to determine if enough bandwidth is available along the nodes in a proposed route. Paper [11] suggests that the admission control scheme take into account the excess amount of resources a call uses. It is possible that the shortest route for a call is congested so that a longer route must be used. In paper [3], the excess amount of resources used is measured by the ratio of the number of hops (nodes) used in a call and the minimal number of hops possible for the call (the *stretch factor*). If the stretch factor is significantly large, then the call should be rejected because a large stretch factor implies that an excessive amount of unnecessary resources are required to use that route. In the long run, if stretch factors are kept small, most calls in the network use a minimal number of hops so that more calls can be admitted to the network.

### 3.2 The vector approach

To determine if a route is congested for call admission, the network must determine at each node if it can accept the new call. Each node in the route must determine if the resources required by the call is available. This is not only to determine if the new call should be accepted, but to update the statuses of each node of the amount of bandwidth used. This can become an expensive process, especially if repeated several times to attempt one call admission. Unnecessary processing should be avoided whenever possible. The following scheme is proposed in [12] to avoid some processing required in admission control. The idea behind this scheme is that with two fast binary operations, we can maintain a table to determine if a proposed route includes a congested link. This scheme can assist in avoiding node-by-node acceptance of calls using routes that are likely to fail due to one or more congested links. This scheme reduces traffic, processing, and call set-up delay.

In this scheme proposed in [12], each node develops a bit vector (the *out vector*) with each bit corresponding to the state of a link, whether or not it is congested. If a link is congested, then its corresponding bit is set to 1, otherwise it is set to 0. The out vector represents the states of all links in the network so that all nodes in the network has some information on all links in the network.

The out vector can be used to determine if a proposed route for a call contains a congested link by first translating the proposed route to a vector form showing which links are used in the route. To determine if a route is congested, the route vector is AND'ed with the out vector. If any of the bits in the result is nonzero, then the route must contain a congested link and an alternate route should be found. The process by which the out vector is produced is described below.

Periodically, say every 500 $\mu$s (on local clocks), the nodes broadcast a message with an out vector on all of its outgoing links. For example, take the network of figure 1. Suppose that node $D$ has a bit vector that indicates links 1, 3, and 7 congested and node $E$ has a vector that indicates

links 1, 5, and 8 congested as shown in the "Links" column of figure 2. When these vectors are broadcasted, they will eventually reach node $F$.

From the perspective of the receiving node, $F$, it should be apparent which of its incoming links that the most updated information about which links arrive from. In this example, suppose that information on links 1, 3, and 4 reach $F$ fastest through $D$ and information on links 2, 5, and 6 reach $F$ fastest through $E$. This information is used to create an AND mask of "who knows best" as shown in figure 2. A fast AND operation with the received vectors and the bit masks produces the most up to date information on the congestion states of links. The masks are used to ensure that only the most up-to-date information is used, since information of a particular link can arrives through several paths. From the perspective of node $F$ in figure 1, it takes longer for information about link 1 to reach node $E$ than it does to reach $D$ or itself. So the mask for $D$ at node $F$ includes link 1 and so forth. Thus, the masks represent which nodes "knows best" about which links.

The resulting received vectors are combined with an OR operation to produce the *in vector*. The in vector represents the states of nodes which are known about better through other nodes. In this example, states of links 1 through 6 are updated faster at nodes $D$ and $E$ than at $F$ because of the proximities to those links. The in vector in this example shows that links 1, 3, and 5 are congested.

Node $F$, the receiving node also finds the congestion states of links it "knows best." Figure 2 shows that link 8 is congested. This vector is OR'ed with the in vector to produce the out vector. The out vector is stored for $F$'s traffic information and broadcasted to nodes $D$ and $E$.

This scheme, although it uses a broadcast method, eliminates heavy processing required to determine if a route contains a congested link. The processing required to maintain the out vectors is also small. This scheme saves processing time by reducing the processing required to a set of quick binary operations. This is extremely important in an ATM environment where processing must be kept to a minimum. The vector approach produces some additional traffic in the network. However, with the large amount of available bandwidth, its effect should be minimal.

The vector approach has some disadvantages. This method is sensitive to the network topology and particularly to changes in the topology. Each node has to know the exact topology of the network and propagation delay times for all links in order to generate the masks of "who knows best" and to translate routes to vector format. Large networks may be infeasible since they require large bit maps and complex maintenance of the network bit masks, particularly if nodes periodically fail or enter the network.

The effectiveness of the vector approach in an environment of large propagation delays is questionable. The status information can be received too late to be up to date. Some form of long-term averaging of the out vectors to obtain a somewhat more accurate description of a link's long term bandwidth usage may be more suitable for admission control.

### 3.3 Resource allocation

The vector approach can quickly determine if a route is likely to be congested. Once an uncongested route has been found, the admission control process must allocate resources for the new call at each node. During this process, a node may determine that it does not have enough resources to support the new call. The schemes described in this subsection can be used to accept or reject new calls based on their expected bandwidth usage.

In the first scheme, admission control is based on a threshold allocation scheme. In this scheme, calls are admitted as long as bandwidth is available (i.e. the equivalent bandwidth of a new call does not push the expected bandwidth usage for a node over the threshold of available bandwidth). This scheme is simple but does not fairly admit calls. Accepting a few calls requiring a large amount of bandwidth can block several smaller calls such that the call blocking probability is unacceptably high. Additionally, this scheme suffers from potentially high bandwidth waste. An inproportionate amount of bandwidth at a particular node may go to a few virtual paths. In addition to potential unfair call blocking, calls may be rejected because of a lack of bandwidth at that node even though other nodes in their paths may have a large amount of free bandwidth. Thus the potential exists for bandwidth to be underutilized.

The second scheme addresses the issue of unfair blocking, blocking several low bandwidth calls for a few high bandwidth calls. Connections with high bandwidth requirements should not consume so much available bandwidth that calls with low bandwidth requirements experience an unacceptable call blocking probability. Paper [13] proposed that a call should be admitted if its bandwidth requirement does not exceed some predetermined percentage, $X$, of available bandwidth. Paper [13] showed through simulation that the arrival rate of new calls from a class with high bandwidth requirements did not affect the call blocking probability of other classes with lower bandwidth requirements. These simulations also showed that to significantly improve call blocking probabilities of low bandwidth calls, $X$ needed to be small, 0.2 to 0.4. Note that $X = 1$ corresponds to threshold allocation of the first scheme described above. The call blocking probability for this scheme in [13] is lower for low bandwidth calls than it is in the threshold scheme at the expense of the call blocking probability for calls with high bandwidth requirements.

The third scheme addresses the issue of having an inproportionate amount of bandwidth allocated to a few virtual paths. A virtual path (v.p.) is a logical link between two nodes established on a long term basis made up of a number of connections. A v.p. has its own allocated bandwidth and thus, has an upper limit on the number of connections the v.p. can support. Paper [14] proposed that each virtual path be allocated some bandwidth to satisfy some minimal requirement. The remaining bandwidth can be allocated to v.p.'s upon demand as their bandwidth requirements exceed minimum allocations. This dynamically allocated bandwidth is released when the virtual path no longer requires it. Although this may cause some bandwidth to be wasted if some virtual paths do not have many connections, analysis of two v.p.'s with multiple connections on a single link showed that call blocking rates decreased and throughput improved by as much as 20% compared to the threshold allocation described in scheme 1. However, allocating bandwidth per virtual

path is a volatile process. There are no mechanisms for handling changes in the network. Also, a lot of bandwidth may remain unused if some v.p.'s do not have many calls but still have allocated bandwidth. Some dynamic flexibility in bandwidth allocation may prove useful.

The disadvantage of the three admission control schemes described above are the amount of memory required and added processing overhead. The processes for "allocating" bandwidth must be performed at each node along a proposed route. Each node must keep an account of the resources it has allocated and process call set-up packets. If a call is rejected, then a message must be sent back through the same route so that the intermediate nodes can free the resources. Each node must also process call tear-downs. This not only increases the design complexity of ATM circuits, but creates additional traffic upon rejection.

For the fourth scheme, paper [12] proposes the use of scout packets to detect congestion along a path to reduce processing required in call set-up. Scouts are pseudo- packets (in [12]'s terminology) which traverse the candidate route to the destination. Scouts are treated like a data packet, not as a control packet. Nodes do not have to perform costly operations on scouts as they would in previous admission schemes. Its priority level is set between those of real-time and non-real-time communications.

To set up a real-time connection, a stream of scout packets with the same traffic characteristics as the desired connection (such as average bit rate and burstiness) is transmitted on the desired route. If the scouts do not find congestion on the path, then they will reach their destination and return to the source within the specified maximum delay. If the scouts find congestion along the path, then the scouts will either arrive at their destination late or be dropped at the congested node(s). In either case, the call is rejected, and an alternate route can be attempted or the same route reattempted after a timeout.

During heavy network load, the fourth scheme using scouts will tend to favor routes with fewer hops since the probability of finding congestion on on fewer hops is smaller. This can help increase throughput since calls with a small number of hops do not tie up resources on as many nodes as calls with more hops [11].

The scout packet method of call admission and set-up may be the most viable approach for ATM among the four schemes described. However, it is statistical in that it does not guarantee that resources are available for admitted calls. It is possible that a large number of already existing calls were idle when the scouts were sent. Thus, a scout method may admit more calls than resources allow resulting in possible resource shortage when idle calls become active. The statistical nature of the scout method should be investigated. Another problem with the scout method is that it creates a lot of excess traffic if a large portion of call set-up attempts are rejected and reattempted later on the same or an alternate route.

Admission control is a complicated issue. Determining equivalent bandwidth is a difficult process at best. It is possible that equivalent bandwidth may need to account for any traffic shaping that will be imposed on the traffic source. In this case, the scheme for determining equivalent bandwidth will need to be made in conjunction with the smoothing scheme used.

# 4 Smoothing Schemes and Performance

To avoid short term congestion caused by bursts of cell transmissions, we can shape the traffic. This section discusses schemes to smooth bursty traffic upon entering the network and also at intermediate nodes to reduce burstiness. Smoothing schemes shapes the incoming traffic reducing the negative effects of burstiness. Smoothing schemes also enforce the bandwidth allocated to a call. If calls often exceed their allocated bandwidths, then the admission control scheme used is inaccurate since it works from inaccurate data. Thus, bandwidth enforcement ensures the validity of admission control schemes based on bandwidth allocation.

## 4.1 Leaky bucket and variants

To shape the incoming traffic at network edges and enforce allocated bandwidth, we can introduce a token pool, also known as a leaky bucket, to the system. In a leaky bucket scheme, cells can only be transmitted when they can obtain a token from a token pool. If the token pool is empty, then the cell must either wait for a token before it is delivered to the next node, or it can be dropped entirely. Tokens are generated at some (constant) rate $r$ and stored in a token pool. The pool has a finite size denoted by $\sigma$. After filling the token pool, tokens arriving to the pool are discarded. $\sigma$ can be seen as the maximum allowable burst length since a maximum of $\sigma$ cells may be delivered at one time. The token generation rate, $r$, is a measure of the average bit rate.

Some feel that the leaky bucket method is unfair [15]. It only allows small variations from the average bit rate allocated to a call. A call cannot transmit at a rate greater than $r$ for more than a brief period of time. This is particularly true in the system described in [16] with no cell buffer. To allow some statistical variation, a much larger bandwidth has to be allocated. However, this reduces some of the advantages of statistical multiplexing. To solve this problem, a virtual leaky bucket was proposed [17,18].

In a virtual leaky bucket, cells arriving to an empty token pool are marked red and transmitted without a token, while those that have tokens are marked green. Red cells are considered violators of allocated bit rate since the call must have exceed the allocated bit rate for some time for the token pool to be empty. Because bandwidth may still be available, marking red cells allows the call to exceed its allocated bit rate if it does not adversely affect other calls. If at some point along its path, a red cell reaches a congested node, then it may be discarded so that the throughput of green cells is not significantly affected. Hence, red cells are given lower priority than green cells.

[17] also proposed the use of a spacer in a virtual leaky bucket to smooth out bursts as shown in figure 3. The spacer is used to control the total maximum rate of red and green cells. In a virtual leaky bucket scheme with spacer, two kinds of tokens are generated: red and green tokens. Red and green tokens are generated at rates $\gamma_g$ and $\gamma_r$, respectively, and inserted into separate token pools. As before, when the green token pool empties, cells can transmitted marked red. However, in the previous scheme the rate at which red cells could be transmitted is unlimited. In this scheme, red

cells are required to obtain red tokens. Having two token pools limits the rate at which red cells are transmitted as well as green cells.

In the conceptual model shown in figure 3, when a cell is delivered at point A, its token is removed and the token enters a spacer. The tokens are discarded from the spacer bank at rate $\beta$. The next cell at the head of the output queue cannot be transmitted even if there are green or red tokens available in the pool unless the spacer is empty. This assures us that cells are transmitted at a rate less than $\beta$ even during bursts and even when both token pools are full by inserting "spaces" between cells. This effectively allows two levels of rate control whereas without a spacer, there is only one level of control, i.e. red cells can be transmitted at will. The rates of green and red cells and maximum burst size of each are controlled individually by controlling $\gamma_g$ and $\gamma_r$. Limiting the transmission of red cells avoids a flood of red cells congesting a later node. Additionally, a spacer allows the network to control the overall transmission rate to a manageable level.

Some have suggested that marking for purposes of bandwidth policing offers no significant advantages [4]. It is difficult to predict the quality of service since red cells may be easily dropped in the network. This, however, may not be a significant disadvantage as long as the performance of green cells (minimum performance requirements) can be guaranteed. Therefore, our next concern is how to guarantee that the performance of green cells will not be affected by red cells. There is an optimal strategy to achieve this. In the optimal strategy, green cells can preemptively discard all red cells, including those in service. In this strategy, there should be no red cells in the buffer or in service if any green cells are being discarded or excessively delayed. In other words, no performance degradation for green cells will be caused by the presence of red cells. However, this requires dropping red cells from arbitrary points in the buffer. This may prove to be extremely difficult, and therefore, we should consider some alternative strategies. In [17], it is shown that a simple, easily implementable threshold strategy can yield the performance close to that of optimal. In this threshold strategy, red cells are accepted only if the queue occupancy is less than a certain threshold. Therefore, authors of this paper believe that with a good strategy to handle green and red cells, we can guarantee that the performance of green cells is not affected by red cells. As long as we can guarantee this, the advantages of marking seem to be unquestionable.

### 4.2 Rate controls

Rate controls regulate the rate at which traffic enters the network like the leaky bucket. Rate controls are based on a smoothing interval of time $T$ and some measure of maximum transmission rate. In a leaky bucket, when a call is idle, tokens accumulate in the token pool. When the call becomes active, it can immediately send a large number of cells. Calls under rate controls do not accumulate transmission times when they remain idle as they do in leaky bucket methods. If the token pool size in a leaky bucket is 1, then tokens do not accumulate when the call is idle since only one token will be stored. Thus, rate controls are similar to leaky buckets where token pools are of size 1.

One example of a rate control is a $(r, T)$-smooth stream proposed in [19]. An input stream is

defined as $(r, T)$-smooth if during each time frame of length $T$, the input stream generates no more than $r \cdot T$ bits. $r$ can be seen as a measure of the average bit rate of the call and $T$, also known as the smoothing interval, as an indication of burstiness.

If the source requires more than $r \cdot T$ bits in one frame time, then the source must wait until the next frame to transmit the excessive traffic (bits generated - $r \cdot T$ bits) causing some added delay locally. Thus imposing $(r, T)$-smoothness on a traffic stream smooths traffic entering the network at the cost of increased delay and occasional cell losses locally (due local buffers overflowing) to maintain overall performance throughout the network. It has been shown to minimize delay variance compared to straightforward non-smoothed traffic. Distributed Source Control (DSC) [20] uses a similar technique.

However, even if cells enter the network smoothly, they can cluster together to form longer and longer bursts at intermediate nodes in the network [21]. We need some way to maintain the original smoothness through intermediate nodes in the network. Stop-and-go queueing has been offered as a possible solution to this problem in [19,21]. In stop-and- go queueing, cells arriving during some frame $F$, which could interpreted as the smoothing interval $T$ described above, do not become eligible for transmission until the next frame, $F + 1$. The following conclusions are made in [19,21]:

1. During one departure frame at each node, the number of cells to be transmitted constantly reduce in size. This is because cells arriving during frame $F$ must wait until frame $F + 1$ to be transmitted such that no new cells can become eligible for transmission during frame $F$.
2. Once a cell is eligible for transmission, it will receive service in $T$ seconds or less where $T$ is the frame length.
3. The traffic stream of each connection will maintain the original smoothness property throughout the network.
4. A buffer space of at most $3C_lT$ per link $l$ is enough to eliminate buffer overflow in the network where $C_l$ is the link rate.
5. Total queueing delay is bound between $H \cdot T$ and $2 \cdot H \cdot T$ where $H$ is the number of hops along the path.

To allow for circuit emulations and to reduce the buffer size, the frame length, $T$, should not be greater than 1 msec [21]. This will maintain the total queueing delay under a few msec. As the smoothing interval $T$ is reduced while keeping the same average bit rate $r$, paper [20] showed that the mean and variance of cell waiting time are reduced proportionately for the three traffic types considered: voice, video, and data. This follows from conclusion 5 above. Smaller frame times also reduce the required buffer sizes from conclusion 4.

However, in ATM networks bandwidth is allocated in cell sizes since cell sizes are fixed and thus, a small frame size increases the step size of bandwidth allocation. Suppose that bandwidth is allocated in increments of $\Delta r$. We then have:

$$\Delta r = \frac{Cell\ size\ [53 \cdot 8\ bits]}{T}$$

For $T = 1$ msec, bandwidth is allocated in increments of 424 kb/s which is much larger than requirements for many services. Thus, it would be beneficial to study feasibility prospects of having different values of $T$ for different classes of service to accommodate various services. However, if this is too complicated for implementation in ATM, queueing delay and capacity allocation requirements can be reasonably compromised to make a single network-wide value of $T$.

Implementation of stop-and-go queueing is quite simple. In figure 4(a), a 2-queue implementation is shown where all cells arriving during some frame $F$ is inserted into one of two queues. While arriving cells are inserted in one queue, all cells to be serviced are obtained from the other queue. During frame $F+1$, the two switches flip to the other queues, and the first queue is serviced and arriving cells are placed in the second queue. In a single queue implementation of stop-and-go queueing, shown in figure 4(b), all cells are placed into a single queue. The service controller records which cell was the last cell to arrive during a frame with a pointer. This pointer also simultaneously marks the first cell of the next frame. Note that the service controller only needs one pointer. After the last cell of a frame is serviced, the service controller switches off the link between the queue and the server until the beginning of the next frame. The service controller then switches on the link between queue and server, enabling cells from the next frame to be serviced.

The 2-queue server with rate control, introduced in [22], can be used in conjunction with an $(r, T)$-smooth traffic stream to provide guaranteed service for connection oriented services such as CBO traffic. To explain this model, let's assume that we only have two classes of traffic, CBO and data. In the 2-queue server of [22], for each byte of data serviced, we explicitly reserve bandwidth for $K$ times as many bytes of CBO traffic. Data traffic has an associated time of service (TOS) which is a measure of its delay requirement per node. Recall that data traffic is reasonably flexible in delay sensitivity. The server services CBO traffic or remains idle if there is no CBO traffic until the time spent since the last data traffic serviced becomes greater than TOS. In this scheme, CBO traffic receives enough bandwidth to meet its delay and loss requirements as long as a good choice of $K$ is made (approximately 3). Data traffic is guaranteed to be serviced at intervals of TOS. The 2-queue server strategy provides good circuit emulation for CBO traffic. Simulations showed that this strategy is as good as a scheme where a fixed priority is given to CBO in circuit source and the size of data served during each TOS interval. The 2-queue server strategy also smooths bursty data by scheduling circuit-like traffic between packets.

The 2-queue server model seems to be best suited for circuit emulation in an ATM network. This strategy requires knowing in advance the amount of CBO traffic relative to data traffic in order to determine the right value for $K$. Studies should be made into possible solutions such as varying $K$ to meet the needs of traffic dynamics.

### 4.3 Observations

We need to control the rate at which cells are delivered during bursts. A simple leaky bucket with or without marking allows large bursts (up to the size of the token pool) with no smoothing interval between cells. Imposing $(r, T)$-smoothness restricts burstiness to a rate of $r$ but does not allow any flexibility beyond that rate for even short periods of time. The leaky bucket scheme with a spacer is effectively the same as $(r, T)$-smoothness but additionally allows marking of cells. The maximum rate is given by $\beta$, the spacer's rate, with an average rate determined by the token generation rate(s) whereas in $(r, T)$- smoothness, the maximum rate and the average rate could both be $r$.

While bursts do need to be controlled, we also need to allow some flexibility from allocated bandwidth while maintaining the long term cell generation rate of calls. While $(r, T)$-smooth streams ensures a good measure of smoothness, it does not allow much in terms of flexibility, nor does a leaky bucket where cells are dropped when token pools are empty. The leaky bucket with spacer flexibly controls bit rates during bursts with two levels of control. Additionally, it allows marking of cells to take advantage of possible available bandwidth without affecting the high priority "green" traffic significantly.

## 5 Conclusion

This paper covered some of the congestion control methods commonly considered to be suitable for an ATM network. We have covered various issues such as characteristics of bursty traffic, bandwidth allocation, call admission, and rate control schemes. The high processing speed changes the focus of network design from bandwidth efficiency to processing efficiency. It also moves the focus from feedback control to a preventive forward control. Schemes that are most likely to succeed in ATM emphasize congestion prevention (such as call admittance, bandwidth policing, and rate control) over reactive congestion control based on feedback.

## References


[1] Steven E. Minzer, "Broadband ISDN and asynchronous transfer mode," IEEE Commun. Mag., vol. 27, no. 9, pp. 17-24, Sep 1989.

[2] Duke Hong and Tatsuya Suda, "Survey of Techniques for Prevention and Control of Congestion in an ATM Network," Tech. Report, U.C., Irvine, 1990.

[3] Israel Cidon and Inder S. Gopal, "Control mechanisms for high speed networks," IEEE ICC 1990, pp. 259-263.

[4] C. Anthony Cooper and Kun I. Park, "Toward a broadband congestion control strategy," IEEE Network Magazine, vol. 4, no. 3, pp. 18-23, May 1990.

[5] Jaime Jungok Bae and Tatsuya Suda, "Survey of traffic control schemes and protocols in ATM networks," to appear in proceedings of the IEEE.

[6] T. Bradley and T. Suda, "Survey of unified approaches to integrated-service networks," Proc. of the IEEE International Telecommunications Symposium, Sept. 1990.

[7] M. Decina, T. Toniatti, P. Vaccari, and L. Verri, "Bandwidth assignment and virtual call blocking in ATM networks," INFOCOM 1990, pp. 881-888.

[8] Maurizio Decina and Tiziana Toniatti, "On bandwidth allocation to bursty virtual connections in ATM networks," IEEE ICC 1990, pp. 844-851.

[9] H. Suzuki, T. Murase, S. Sato, and T. Takeuchi, "A burst control strategy for ATM networks," C&C Systems Research Laboratories, NEC Corp., to appear in GLOBECOM 1990.

[10] Nanying Yin, San-qi Li, and Thomas E. Stern, "Congestion control for packet voice by selective packet discarding," IEEE Trans. Commun., vol. 38, no. 5, pp. 674-683, May 1990.

[11] K. Noguchi, T. Okada, and H. Ohnishi, "Resource management in an ATM network," 2nd IEEE COMSOC International Multimedia Communications Workshop, Montebello, Quebec, Canada, April 1989.

[12] M. Decina, T. Toniatti, P. Vaccari, and L. Verri, "Bandwidth assignment and virtual call blocking in ATM networks," INFOCOM 1990, pp. 881-888.

[13] M. Ilyas and H.T. Mouftah, "Performance evaluation of congestion avoidance in broadband ISDNs," IEEE ICC 1990, pp. 727-731.

[14] Weilin Wang, Tarek N. Saadawi, and Ken-ichi Aihara, "Bandwidth allocation for ATM networks," IEEE ICC 1990, pp. 439-442.

[15] G. Gallassi, G. Risolio, and L. Verri, "Resource management and dimensioning in ATM networks," IEEE Network Magazine, vol. 4, no. 3, pp. 8-17, May 1990.

[16] Arthur W. Berger, "Performance analysis of a rate control throttle where tokens and jobs queue," INFOCOM 1990, pp. 30-38.

[17] Krishna Bala, Israel Cidon, and Khosrow Shoraby, "Congestion control for high speed packet switched networks," IEEE INFOCOM 1990, pp. 520-526.

[18] G. Gallassi, G. Rigolio, and L. Fratta, "ATM: Bandwidth assignment and bandwidth enforcement policies," GLOBECOM, Dallas, Texas, 1989.

[19] S. Jamaloddin Golestani, "Congestion-free communication in broadband packet networks," IEEE ICC 1990, pp. 489-494.

[20] G. Ramamurthy and R.S. Dighe, "Distributed source control: A network access control for integrated broadband packet networks," INFOCOM 1990, pp. 896-907.

[21] S. Jamaloddin Golestani, "Congestion-free transmission of real-time traffic in packet networks," IEEE INFOCOM 1990, pp. 527-536.

[22] R.S. Dighe, C.J. May, and G. Ramamurthy, "Congestion avoidance strategies in broadband packet networks," submitted to INFOCOM 1991.

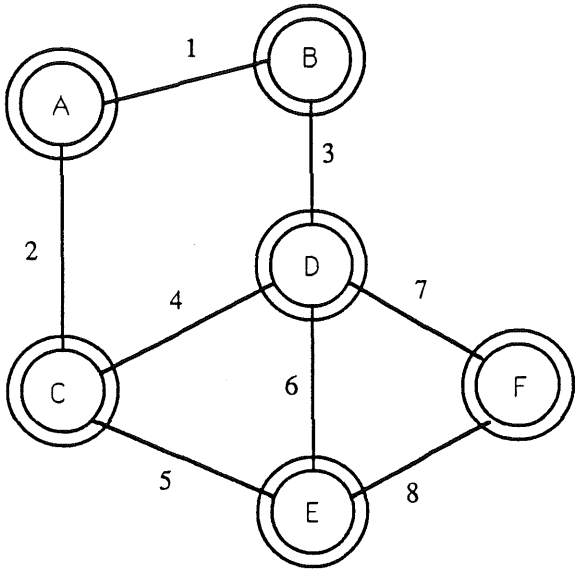


Figure 1

Network example for vector approach

| *Received from* | *Links* (1 = congested) | *Mask* (knows best) | *Logical* AND |
|---|---|---|---|
| node *D* | 10100010 | 10110000 | 10100000 |
| node *E* | 10001001 | 01001100 | 00001000 |
| | | In Vector | 10101000 |
| | | Status from node *F* | .....01 |
| | | Out Vector | 10101001 |

To Be Broadcast

Figure 2

Congestion vectors for node *F*

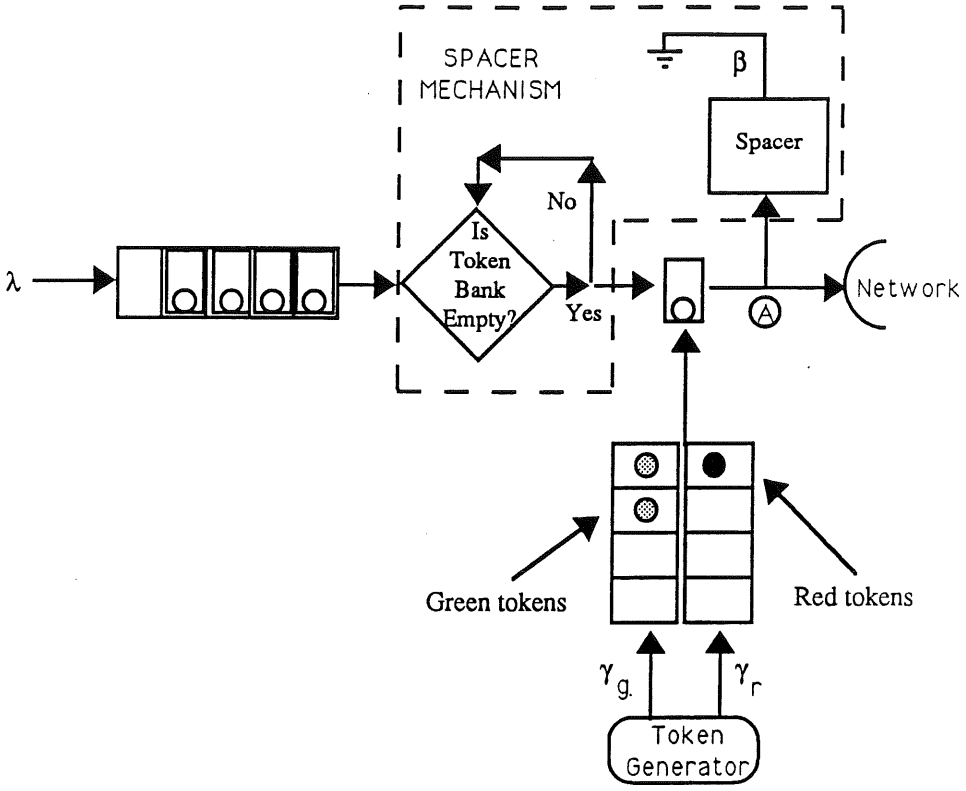


Figure 3

Virtual Leaky Bucket with Spacer

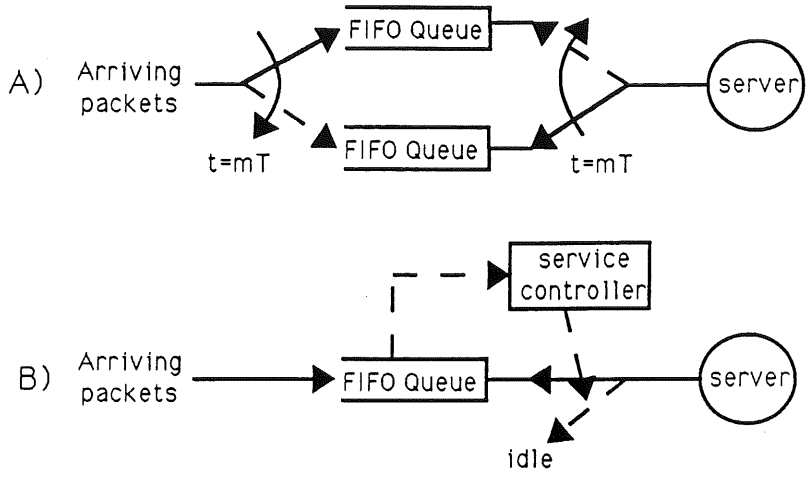


Figure 4
Two realizations of stop-and-go queueing strategy
a) A double-queue structure
b) A single-queue structure